A Semantic Analysis of Wireless Network Security Protocols

Massimo Merro

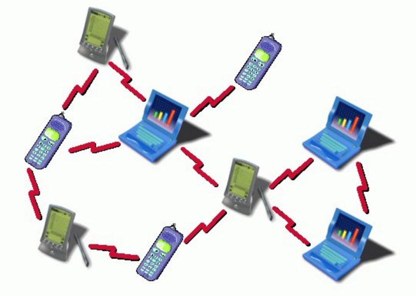
(joint work with Damiano Macedonio)

Dipartimento di Informatica

Università degli Studi di Verona - Italy

NFM 2012 - Norfolk, April 3-5 2012

Wireless Networks (Ad hoc, Sensor, Vehicular, Mesh networks . . .)



- **Some challenging features**:
  - No fixed infrastructure
  - Radio frequency channels
  - Half-duplex channels
  - Local broadcast
  - Multi-hop communication
  - High vulnerability

Attacking and securing wireless networks

- In a wireless network an attacker may:
  - compromise a node (node subversion)
  - alter data integrity
  - eavesdrop on messages
  - inject fake messages
  - waste network resources
  - etc
- Designing security protocols for wireless networks requires a deep understanding of their resource limitations (Processor, Memory, Battery power, etc)

A process algebraic approach to model wireless networks

Assumptions:

- Synchronisation: all nodes are synchronised using a *clock-correction* synchronisation protocol (this implies network connectivity)
- Time: proceeds in *discrete steps*; a global clock is supposed to be updated whenever all nodes agree on this, by synchronising on a special action σ ([Hennessy and Regan 1995])
- Fictitious clock approach: data transmission is assumed to take no time. This is reasonable if the actual time of transmission is negligible with respect to our time intervals
- Nondeterminism: untimed activitivies among nodes occur nondeterministically
- Mobility: Our nodes are stationary (as in most sensor networks); communication and node mobility are orthogonal concepts

The Syntax

Networks :

$$
\begin{array}{llll}
M, N & ::= & \mathbf{0} & \text{empty network} \\
 & \mid & M_1 \mid M_2 & \text{parallel composition} \\
 & \mid & n[\![P]\!]^{\nu} & \text{node } \ (\nu = \text{set of neighbours of } n)
\end{array}
$$

Processes :

$$
\begin{array}{llll}
P, Q, R & ::= & \mathsf{nil} & \text{termination} \\
 & \mid & \sigma.P & \text{sleep} \\
 & \mid & !\langle u \rangle.P & \text{broadcast} \\
 & \mid & \lfloor ?(x).P \rfloor Q & \text{receiver with timeout} \\
 & \mid & \lfloor \sum_{i \in I} \tau.P_i \rfloor Q & \text{internal choice with timeout} \\
 & \mid & [u_1 \ldots u_n \vdash_{\mathrm{r}} x]P; Q & \text{deduction}
\end{array}
$$

- The calculus is parametric wrt to a given *decidable* inference system

Labelled Transition Semantics (some rules)

$$(\text{Snd})\ \frac{-}{m[!\langle v\rangle.P]^{\nu} \xrightarrow{m!v\triangleright\nu} m[P]^{\nu}}$$

$$(\text{Rcv})\ \frac{m \in \nu}{n[\lfloor ?(x).P \rfloor Q]^{\nu} \xrightarrow{m?v} n[\{{}^{v}\!/\!{}_{x}\}P]^{\nu}}$$

$$(\text{Bcast})\ \frac{M \xrightarrow{m!v\triangleright\nu} M' \quad N \xrightarrow{m?v} N' \quad \mu := \nu\backslash \text{nds}\,(N)}{M \mid N \xrightarrow{m!v\triangleright\mu} M' \mid N'}$$

$$(\text{Sleep})\ \frac{-}{n[\sigma.P]^{\nu} \xrightarrow{\sigma} n[P]^{\nu}} \qquad (\text{Timeout})\ \frac{-}{n[\lfloor ?(x).P \rfloor Q]^{\nu} \xrightarrow{\sigma} n[Q]^{\nu}}$$

$$(\text{TimeSync})\ \frac{M \xrightarrow{\sigma} M' \quad N \xrightarrow{\sigma} N'}{M \mid N \xrightarrow{\sigma} M' \mid N'}$$

Simulation theory

We are interested in a weak semantics which abstracts over internal actions, $\xrightarrow{\tau}$

Weak transitions

They are defined as usual:

- $\xRightarrow{\hat{\alpha}} \stackrel{\text{def}}{=} \xrightarrow{\tau}^{*} \xrightarrow{\alpha} \xrightarrow{\tau}^{*}$, if $\alpha \neq \tau$
- $\xRightarrow{\hat{\tau}} \stackrel{\text{def}}{=} \xrightarrow{\tau}^{*}$

Definition: Similarity

- $M \lesssim N$ if $M \xrightarrow{\alpha} M'$ implies $\exists N'$ s.t $N \xRightarrow{\hat{\alpha}} N'$ and $M' \lesssim N'$

Theorem: Pre-congruence result

The binary relation $\lesssim$ is a congruence over networks

Adapting tGNDC to wireless networks

Gorrieri and Martinelli's tGNDC is a general framework for the formal verification of security properties in a concurrent scenario. Intuitively:

A protocol M satisfies $tGNDC^{\rho(M)}$ if the presence on an arbitrary attacker does not affect M wrt the chosen abstraction $\rho(M)$ of the protocol.

tGNDC more formally:

A protocol M satisfies $tGNDC^{\rho(M)}$ if for any attacker A it holds that:

$$M \;\Big|\; A \;\lesssim\; \rho(M)$$

Timed security properties:

By varying ρ we can express different timed security properties:

- timed integrity: freshness of authenticated packets
- timed agreement: agreement must be reached within a deadline

A sound proof technique for tGNDC

Proving that a protocol is tGNDC wrt some abstraction requires an universal quantification on all possible attackers. The proof is hard!

Definition: Top attacker

A^{TOP} denotes the Dolev-Yao attacker that can listen (and possibly replay) any message of the protocol. As usual it cannot guess secrets before they are disclosed

Theorem: Criterion for tGNDC

$$M \mid A^{\mathrm{TOP}} \lesssim \rho(M) \quad \text{implies} \quad M \mid A \lesssim \rho(M), \text{ for any } A$$

On the other hand, for proving that a protocol is not tGNDC it is sufficient to exhibit an attacker A and an execution trace for $M \mid A$ which cannot be mimicked by $\rho(M)$ (simulation semantics $\subseteq$ trace semantics)

A case study: The LiSP protocol

- LiSP is a key mangement protocol for Wireless Sensor Networks
- A LiSP network consists of a *Key Server* (KS) and a set of *nodes* $m_1, \ldots, m_k$
- The transmission time is split into time intervals $\Delta_{\mathrm{refresh}}$ long
- The protocol employs two different key families:
  - *master keys* $k_{\mathrm{KS}:m_j}$, one for each node m_j, for initial setup between m_j and BS
  - *temporal keys* $k_0, \ldots, k_n$ used by all nodes to encrypt/decrypt data packets
- Temporal key k_i is tied to time interval i and renewed every $\Delta_{\mathrm{refresh}}$
- At interval i, k_i is shared by all nodes and it is used for encryption

Our Security Analysis: key freshness

Timed integrity requirement for LiSP

- A node should authenticate only keys sent by $\textsc{ks}$ in the last Δ_{refresh} time units
- In fact, if a node would authenticate an obsolete key (older than Δ_{refresh}) then it would not be synchronised with the rest of the network!

The LiSP specification (Key Server)

D_0	$\stackrel{\text{def}}{=}$	$\sigma.D_1$	synchronise and move to D_1
D_i	$\stackrel{\text{def}}{=}$	$[k_i\ k_{s+i} \vdash_{\mathrm{enc}} t_i]$	for $i \geq 1$, encrypt k_{s+i} with k_i
		$[\mathsf{UpdateKey}\ t_i \vdash_{\mathrm{pair}} u_i]$	build the UpdateKey packet u_i
		$!\langle u_i \rangle.\sigma.\sigma.D_{i+1}$	broadcast r_i, and move to D_{i+1}
L_i	$\stackrel{\text{def}}{=}$	$\lfloor ?(r).I_{i+1} \rfloor \sigma.L_{i+1}$	wait for request packets
I_i	$\stackrel{\text{def}}{=}$	$[r \vdash_{\mathrm{fst}} r_1] I_i^1; \sigma.\sigma.L_i$	extract first component
I_i^1	$\stackrel{\text{def}}{=}$	$[r_1 = \mathsf{RequestKey}] I_i^2; \sigma.\sigma.L_i$	check if r_1 is a RequestKey
I_i^2	$\stackrel{\text{def}}{=}$	$[r \vdash_{\mathrm{snd}} m]$	extract node name
		$[k_{\mathrm{KS}:m}\ k_{s+i} \vdash_{\mathrm{enc}} w_i]$	encrypt k_{s+i} with $k_{\mathrm{KS}:m}$
		$[k_{s+i} \vdash_{\mathrm{hash}} h_i]$	calculate hash code for k_{s+i}
		$[w_i\ h_i \vdash_{\mathrm{pair}} r_i]$	build a pair r_i
		$[\mathsf{InitKey}\ r_i \vdash_{\mathrm{pair}} q_i]$	build a InitKey packet q_i
		$\sigma.!\langle q_i \rangle.\sigma.L_i$	broadcast q_i, move to L_i

The LiSP Protocol (receiver at node m)

Z	$\stackrel{\text{def}}{=}$	$[\mathsf{RequestKey}\ m \vdash_{\mathrm{pair}} r]$ $!\langle r\rangle.\sigma.\lfloor ?(q).T \rfloor Z$	send a RequestKey packet wait for a reconfig. packet
T	$\stackrel{\text{def}}{=}$	$[q \vdash_{\mathrm{fst}} q'] T^1; \sigma.Z$	extract fst component of q
T^1	$\stackrel{\text{def}}{=}$	$[q' = \mathsf{InitKey}] T^2; \sigma.Z$	check if q is a InitKey packet
T^2	$\stackrel{\text{def}}{=}$	$[q \vdash_{\mathrm{snd}} q'']$ $[q'' \vdash_{\mathrm{fst}} w] T^3; \sigma.Z$	extract snd component of q extract fst component of q''
T^3	$\stackrel{\text{def}}{=}$	$[q'' \vdash_{\mathrm{snd}} h]$ $[k_{\mathrm{KS}:m}\ w \vdash_{\mathrm{dec}} k] T^3; \sigma.Z$	extract snd component of q'' extract the key
T^4	$\stackrel{\text{def}}{=}$	$[k \vdash_{\mathrm{hash}} h'][h = h'] T^5; \sigma.Z$	verify hash codes
T^5	$\stackrel{\text{def}}{=}$	$\sigma.\sigma.R\langle F^{s-1}(k), k, s{-}1\rangle$	synchronise and move to R
$R(k_{\mathrm{C}}, k_{\mathrm{L}}, l)$	$\stackrel{\text{def}}{=}$	$\lfloor ?(u).E \rfloor F$	wait for incoming packets
E	$\stackrel{\text{def}}{=}$	$[u \vdash_{\mathrm{fst}} u'] E^1; \sigma.F$	extract fst component of u
E^1	$\stackrel{\text{def}}{=}$	$[u' = \mathsf{UpdateKey}] E^2; \sigma.F$	check UpdateKey packet
E^2	$\stackrel{\text{def}}{=}$	$[u \vdash_{\mathrm{snd}} u'']$ $[k_{\mathrm{C}}\ u'' \vdash_{\mathrm{dec}} k] E^3; \sigma.F$	extract snd component of u decrypt u'' by using k_{C}
E^3	$\stackrel{\text{def}}{=}$	$[F^{s-l}(k) = k_{\mathrm{L}}] E^4; \sigma.F$	authenticate k
E^4	$\stackrel{\text{def}}{=}$	$\sigma.\sigma.R\langle F^{s-1}(k), k, s{-}1\rangle$	synchronise and move to R
F	$\stackrel{\text{def}}{=}$	$[l = 0] Z; \sigma.R\langle F^{l-1}(k_{\mathrm{L}}), k_{\mathrm{L}}, l{-}1\rangle$	check if buffer key is empty

Verifying timed integrity 1/2

- The LiSP protocol, in its initial configuration, can be represented as:

$$\mathrm{LiSP} \stackrel{\mathsf{def}}{=} \prod_{j \in J} m_j[\sigma.Z]^{\nu_{m_j}} \mid \mathrm{KL}[\sigma.L_0]^{\nu_{\mathrm{KL}}} \mid \mathrm{KD}[\sigma.D_0]^{\nu_{\mathrm{KD}}}$$

where $m_j \in \nu_{\mathrm{KD}} \cap \nu_{\mathrm{KL}}$ and $\{\mathrm{KD}, \mathrm{KL}\} \subseteq \nu_{m_j}$

- For our analysis it is sufficient to consider only a part of it

$$\mathrm{sLiSP} \stackrel{\mathsf{def}}{=} m[\sigma.Z]^{\nu_m} \mid \mathrm{KL}[\sigma.L_0]^{\nu_{\mathrm{KL}}}$$

Definition: Timed integrity abstraction for sLiSP

$$\rho(\mathrm{sLiSP}) \stackrel{\mathsf{def}}{=} m[\sigma.\widehat{Z}]^{\nu_m} \mid \mathrm{KL}[\sigma.\widehat{L_0}]^{\nu_{\mathrm{KL}}}$$

for appropriate $\widehat{Z}$ and $\widehat{L_0}$.

Proposition: The abstraction is adequate

In $\rho(\mathrm{sLiSP})$ key authentication occurs every $\Delta_{\mathrm{refresh}}$ time units

Verifying timed integrity 2/2

Theorem: Replay attack to LiSP

There is an attacker A such that

$$\mathrm{sLiSP} \mid A \not\lesssim \rho(\mathrm{sLiSP}) \ .$$

Proof Give a trace of $\mathrm{sLiSP} \mid A$ which cannot be matched by $\rho(\mathrm{sLiSP})$!

m	$\longrightarrow$ KL	:	r	m sends a RequestKey to KL
KL	$\longrightarrow m$	:	q_1	KL replies an InitKey lost by m and grasped by A
	$\longrightarrow$			after $\Delta_{\mathrm{refresh}}$ time units
m	$\longrightarrow$ KL	:	r	m sends a new RequestKey which gets lost
A	$\longrightarrow m$	:	q_1	A replays the InitKey q_1 to m
	$\longrightarrow$			after $\Delta_{\mathrm{refresh}}$ time units
m	$\rightarrow *$	:	auth_1	m authenticates the obsolete InitKey q_1

m has authenticated an InitKey which is $2\Delta_{refresh}$ old!!!

$\square$

Can we fix the problem?

Sure! By adding nonces in communications as in other security protocols

Let nsLiSP be the variant of sLiSp with nonces

Theorem: Timed integrity of nsLiSP

For any attacker A

$$\mathrm{nsLiSP} \;\big|\; A \;\lesssim\; \rho(\mathrm{nsLiSP}) \; .$$

Is the protocol with nonces safe now?

Well... when trying to prove *timed agreement* we found a different replay attack (for details see the full paper)

Conclusions

- We have proposed a process calculus to model wireless network security procols
- The calculus comes with both an operational semantics and a simulation theory
- We have adpated Gorrieri and Martinelli's tGNDC to wireless systems
- Provided a soundness criterion for tGNDC
- Analysed the LiSP protocols and found a replay attack on key authentication
- .... and fixed the problem
- Can we use our technique to analyse other protocols? Yes, in the full paper we have applied our tGNDC to analyse both μTESLA and *LEAP+* (here we found another replay attack)